ML Lab Report
Course: UE23CS352A: MACHINE LEARNING
Title: Artificial Neural Networks
Week-3

Name: Chandan Chatragadda	SRN: PES2UG23CS141	Section: 5C

INTRODUCTION

1. Purpose of the lab:
The purpose of this lab was to gain hands-on experience in building a neural network from scratch for function approximation, without using high-level libraries like TensorFlow or PyTorch.

2. Tasks performed:

- Generated a synthetic polynomial dataset using my student ID using the already provided code.
- Implemented neural network components including ReLU activation, MSE loss, forward pass, and backpropagation.
- Split the data into 805 training and 20% testing and trained the neural network with Xavier initialization and gradient descent.
- Evaluated model performance using training/test loss, residuals, and prediction accuracy.

Dataset Description

1. Type of polynomial assigned:
Cubic polynomial: y=2.44x^3 − 0.47x^2 + 3.13x + 9.69

2. Number of samples, features, noise level:

- **Samples:** 100,000 total (80,000 training, 20,000 testing).
- **Features:** 1 input feature (x), 1 output (y).
- **Noise:** Gaussian noise e ~ N(0,2.21).

Methodology

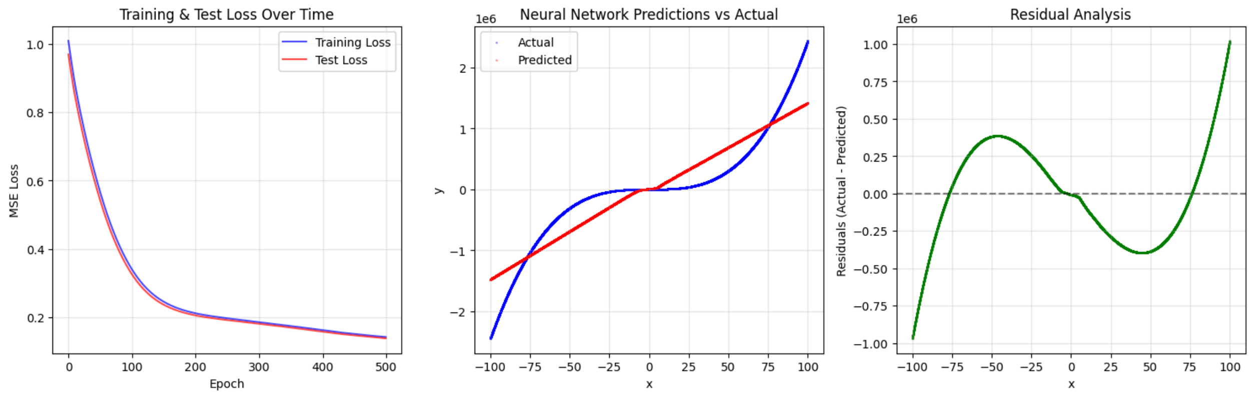
- A synthetic dataset was generated from the assigned polynomial with added Gaussian noise.
- Data was standardized using StandardScaler for both input and output variables.
- The neural network architecture used was: Input(1) → Hidden(32) → Hidden(72) → Output(1).
- Xavier initialization was applied to all weights, with biases set to zero.
- The forward pass applied linear transformation and ReLU activations, followed by a

linear output layer.

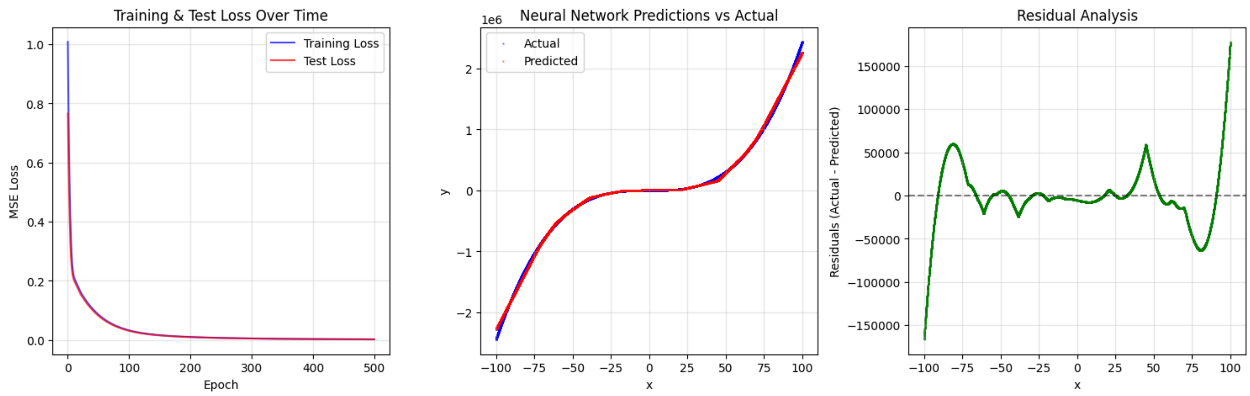
- Backpropagation was implemented manually to compute gradients and update weights via gradient descent.
- Training loop ran for 500 epochs with early stopping (patience = 10).
- Performance was evaluated using MSE loss, residual plots, predicted vs. actual plots, and R^2 score.

Results and Analysis

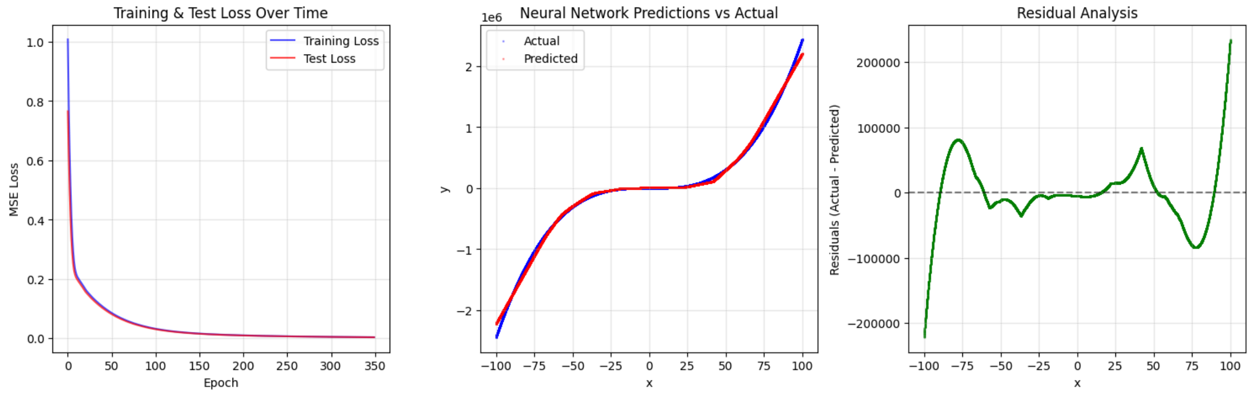
Exp1 (BaseLine)



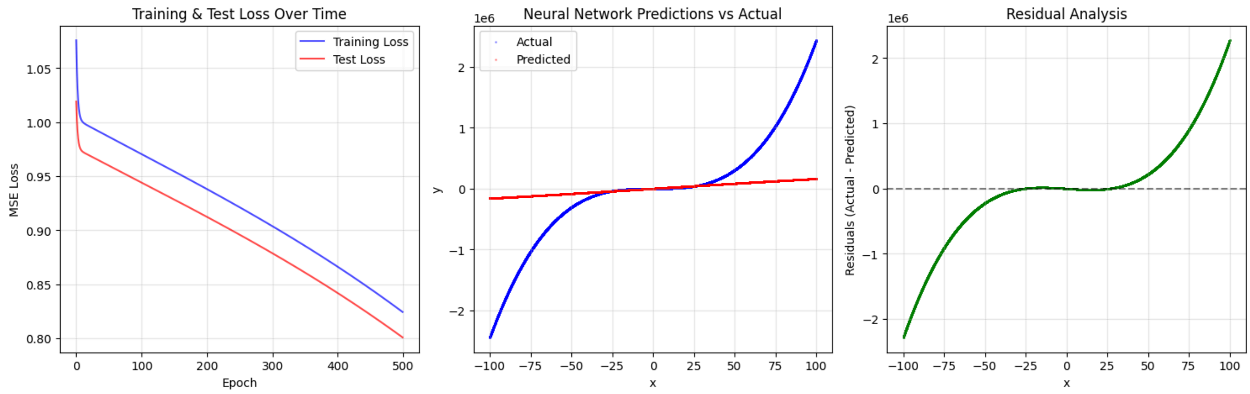
Exp2



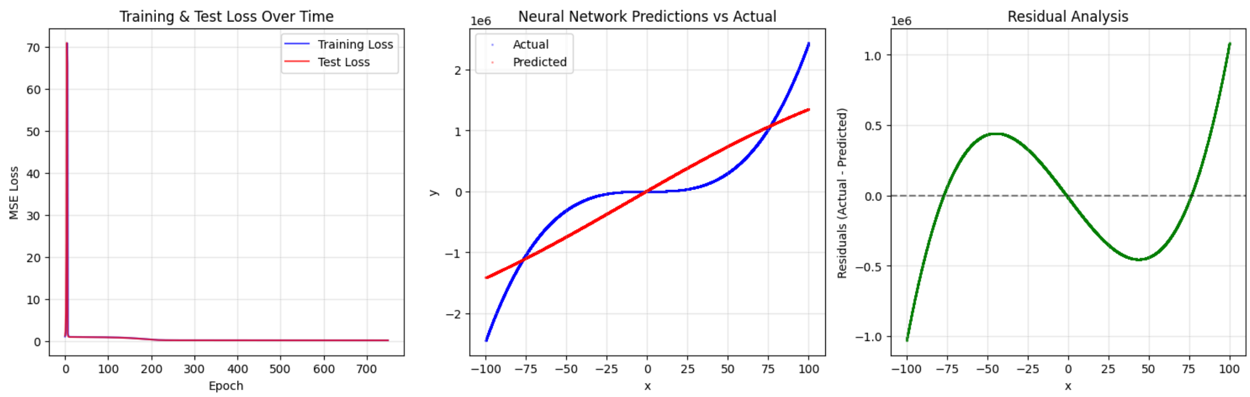
Exp 3


Training & Test Loss Over Time
Training Loss
Test Loss
MSE Loss
Epoch
Neural Network Predictions vs Actual
1e6
Actual
Predicted
y
x
Residual Analysis
Residuals (Actual - Predicted)
x

Exp 4


Training & Test Loss Over Time
Training Loss
Test Loss
MSE Loss
Epoch
Neural Network Predictions vs Actual
1e6
Actual
Predicted
y
x
Residual Analysis
1e6
Residuals (Actual - Predicted)
x

Exp 5


Training & Test Loss Over Time
Training Loss
Test Loss
MSE Loss
Epoch
Neural Network Predictions vs Actual
1e6
Actual
Predicted
y
x
Residual Analysis
1e6
Residuals (Actual - Predicted)
x

Experiment	Learning Rate	No. of epochs	Optimizers	Activation Function	Final Training Loss	Final Test Loss	R^2 Score
Exp1 (Base)	**0.005**	**500**	**Gradient Descent**	**ReLu**	**0.142655**	**0.138746**	**0.8575**
Exp2	**0.095**	**500**	**Gradient Descent**	**ReLu**	**0.001556**	**0.001481**	**0.9985**
Exp3	**0.095**	**350**	**Gradient Descent**	**ReLu**	**0.003228**	**0.003067**	**0.9968**
Exp4	**0.005**	**500**	**Gradient Descent**	**Sigmoid**	**0.824169**	**0.800634**	**0.1774**
Exp5	**0.075**	**750**	**Gradient Descent**	**Sigmoid**	**0.178014**	**0.173953**	**0.8213**

Final Test MSE

From the table:

- The best performing model (Exp2) achieved a Final Test MSE of 0.001481 with an R^2 of 0.9985.
- This indicates the trained neural network almost perfectly approximated the target cubic polynomial with noise.

Base Model (Exp1 – ReLU, lr=0.005): The model learned reasonably well but had relatively higher loss and a modest R^2 of 0.8575. This suggests underfitting, as the network could not fully capture the cubic polynomial with a small learning rate.

Exp2 & Exp3 (ReLU, lr=0.095): Both runs achieved extremely low losses and very high R^2 (>0.996). The close match between training and test loss shows that the model generalized well and avoided overfitting. This configuration is the most effective for the dataset.

Exp4 (Sigmoid, lr=0.005): Performance degraded severely (R^2=0.1774). The Sigmoid activation caused vanishing gradients, leading to underfitting where it failed to approximate the polynomial.

Exp5 (Sigmoid, lr=0.075): Performance improved compared to Exp4, but still lagged behind ReLU models. With R^2=0.8213, it still showed signs of underfitting, though it performed better due to the higher learning rate.

Conclusion

This lab demonstrated how to build a neural network from scratch to approximate a cubic polynomial. The results showed that ReLU activation with a higher learning rate gave the best performance, while Sigmoid often led to underfitting. The best model achieved a very low test MSE (0.001481) and an R^2 of 0.9985, showing it could predict the polynomial almost perfectly.